BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

EXHIBIT A 11 9/25/98

Single wire signaling of isochronous request types in a half-duplex bus

Randy Osborne, MAP
September 24, 1998

1. Problem statement

Since the number of pins is a major factor in the cost of inter-chip connects, there is strong pressure towards making such interconnects fast and narrow. One such example is Hub Link, an I/O interconnect designed by PCD for connecting I/O hubs and PCI bridges (e.g. south bridges) to the memory controller hub (e.g. north bridge). Hub Link is a half-duplex bus with a distributed arbiter. Synchronization occurs via a global clock and two REQ signals, one signaling end A's request to end B and the other signaling end B's request to end A. Each end executes the same algorithm, asserting its REQ signal to convey its request to the remote end, sampling the REQ signal driven by the remote end, and then choosing which end to grant based on the local and remote requests.

There are two major classes of traffic in today's chipsets: asynchronous traffic, in which each data must eventually be delivered, and isochronous traffic, in which each data must be delivered within a specified time. Whenever these traffic classes compete for a shared resource, such as Hub Link, the quality of service received by isochronous traffic can generally be improved if the arbitration for the shared resource is aware of the class of pending requests. In the current version of Hub Link the arbiter cannot distinguish if a pending request, as indicated by sampling REQ asserted, is isochronous or asynchronous. Consequently the Hub Link arbiter must either:

a. assume that all requests are asynchronous, which results in a poor latency bound if a pending request is in fact isochronous e.g. granting several requests from one end before turning the grant over to the other end, or
b. assume that all requests are isochronous and alternating the bus between each end on a request by request basis. This yields a tighter latency bound for isochronous performance, but degrades the asynchronous performance because of the fixed cost of bus turnarounds.

The solution to this dilemma is to add extra information to distinguish isoc and async requests.

2. Prior Art

The most obvious solution is add another pair of wires, one in each direction indicating if that end's request is isoc or async. However, this adds 2 pins to both chips at either end of Hub Link. This is undesirable in a cost-sensitive market such as desktops.

John Garney and Jill Hubbard of PAL have proposed an elegant scheme in which the request type is encoded onto the REQ signal. This scheme requires no additional pins. However, the REQ signal must be decoded at the destination. In an idle system this adds 2 clocks of latency to an asynchronous request from the Least Recently Served end and adds 1 clock of latency to an asynchronous request from the Most Recently Served end. In the case of contention, the REQ decode latency can be overlapped with the current grant and thus the additional latency can be hidden. Thus unless the Hub Link is heavily loaded, asynchronous requests will see additional

latency of 1 or 2 clocks even if isoc traffic is never used. This is unattractive for a server environment in which the I/O read latency is an important factor in system performance.

3. Description of Invention

The subject invention provides the extra information to distinguish isoc and async requests by adding a single signal between both ends of the Hub Link. This adds only 1 pin to each chip on the end of Hub Link. Furthermore, there is no performance impact to asynchronous traffic in the absence of isoc traffic. In an idle system, the latency for either an isochronous or an asynchronous request is exactly the same as in the current, unmodified Hub Link.

The main idea is to convey the type (isoc or async) of a new request at end A to the grant, if any, in progress at the opposite end B. If a grant is in progress, end B compares the new request type with the type of the current grant and then B either keeps asserting REQ or disasserts REQ at the next convenient point, allowing A to proceed and thus affecting a pre-empt. If no grant is in progress at B, then A can be granted.

It suffices to communicate the request type to the opposite end in two circumstances: 1. when a new request is presented while the opposite end is granted, and 2. if idle and MRS end presents a request. When the Hub Link is idle, the LRS end provisionally has the grant – the grant is the LRS's end to lose (which occurs if a REQ arrives from the MRS end and no request from the LRS end claims it). Thus the LRS end does not need to communicate the request type. However, the MRS end must communicate the request type since the LRS end could get the grant between the MRS end sampling the REQ signals and asserting its REQ. In case of a tie between LRS and MRS ends, i.e. simultaneous presentation of requests at both ends, the LRS end wins the grant, even if the LRS end request is async and the MRS end request is isoc.

The subject invention encompasses multiple ways to implement the isoc/async wire:

1. As a wired-OR signal – The wired-OR signal permits the request type communication via a single wire without extra complexity of managing the ownership, and hence turnaround, of the wire.
2. As an alternating ownership wire – This is a conventional common clock signal, such as Hub Link's STOP or PAR, using the same electricals and principle of managed wire ownership. The isoc signal can either be pulsed, where it is active for only one clock when an isoc requets is presented, or it can be driven for the duration a request waits until being granted.

3.1 Electrical issues

The wired-OR signal can be implemented using GTL drivers and parallel termination. However, without the pMOS kicker of GTL+, the rise time is worse than for GTL+. The solution is to constrain the protocol to allow an extra clock (or possibly two clocks if necessary) for ISOC# rise time. See Figures 1 through 6.

The alternating ownership wire can be implemented in either pulsed or duration form for either serial or parallel termination. However, the pulsed version is attractive for parallel termination due to lower power consumption than the duration version. In addition, as explained in Section 3.2.2, the combination of pulsed version and parallel termination permits a slight performance optimization and implementation simplification.

3.2 Protocol details

The following describes the isoc signaling protocol. All three versions, wired-OR, pulsed, and duration are described using the same base protocol followed by appropriate special cases. The isoc wire is named ISOC. The phrase "assert ISOC" in the following is interpreted relative to the protocol version as follows:

- Wired-OR version: "assert ISOC" means assert ISOC# for one clock
- Pulsed version: "assert ISOC" means assert ISOC for one clock
- Duration version: "assert ISOC" means assert ISOC until ownership of the ISOC wire is turned around.

Notation: LRS(A) means that end A is least recently served;
MRS(B) means that end B is most recently served.
RQA(t) means a request is presented by end A at time t
RQB(t) means no request is presented by end B at time t.

1. If idle, i.e. no grant in progress at clock edge k:

If LRS(A)
  If A has a request:
    Assert RQA
    if RQB(k) grant A, i.e. A drives PD and A becomes MRS
    if RQB(k) grant B
  else i.e. no request from A
    if RQB(k) grant B

If MRS(B)
  If B has a request:
    Assert RQB
    If RQA(k) grant B, i.e. B drives PD, and assert ISOC
    If RQA(k) grant A, i.e. A drives PD and A becomes MRS
  Else i.e. no request from B
    If RQA(k) grant A

2. If non-idle at clock edge k: i.e. RQA(k) (Choose end A without loss of generality)

B:
  If RQB(k-1) and B has a request
    Assert RQB and assert ISOC

A:
  If RQB(k-1) and RQB(k) and ISOC asserted at edge k, end B has presented an isoc request.
    If end A is async: disassert RQA at earliest convenient point
    If end A is isoc: continue grant to A (perhaps pre-empt for fairness, but that is outside the scope here).

Additional points:

- A grant can be upgraded to isoc. I.e. during a grant held by end A, that end can serve async and isoc transfers

R Thore 9/24/98

- Timeslice counter suspended while isoc request (so isoc can extend timeslice)
- If end A has grant and samples ISOC asserted during a async transfer, end A can relinquish its grant (at next convenient point), thereby allowing B's isoc request to pre-empt
- Flow control, including the flow control signal STOP, is orthogonal from the wired-OR isoc signaling.

Special cases:

1. Wired-OR version: It is forbidden to sample ISOC# during rise time period (either one or two clock edges after sample ISOC# asserted; depending on the rise time period – diagrams below show one clock for rise time period). However, an exception arises in conjunction with Figure 6 below: ISOC# may be sampled at the same clock edge that a request is asserted (from idle).
2. Pulsed and duration versions: Ownership of the ISOC wire is turned around one clock after the current grant, if any, terminates. If another grant (for the opposite end) begins at that point (i.e. one clock after the previous grant terminates), then assertion of ISOC – if the original end now presents an isoc request – must be delayed for one clock to permit turnaround of the ISOC wire. See the discussion following Figure 12.

3.2.1 Wired-OR Version

The following Figures illustrate the wired-OR version of the protocol.

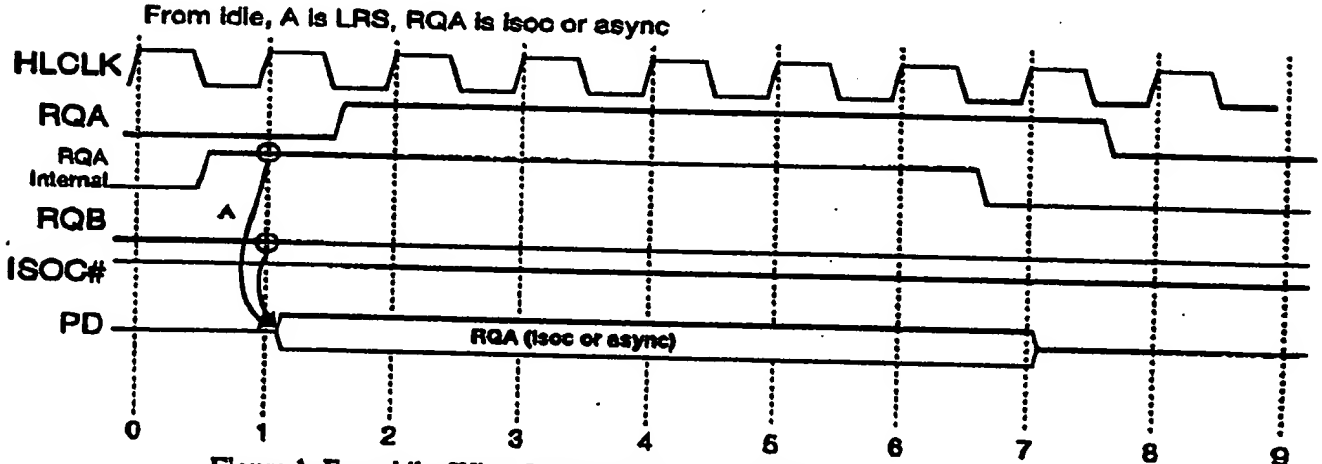

Figure 1: From idle. When LRS and idle, assert RQA; no need to assert ISOC#

R. Ulmer
9/24/98

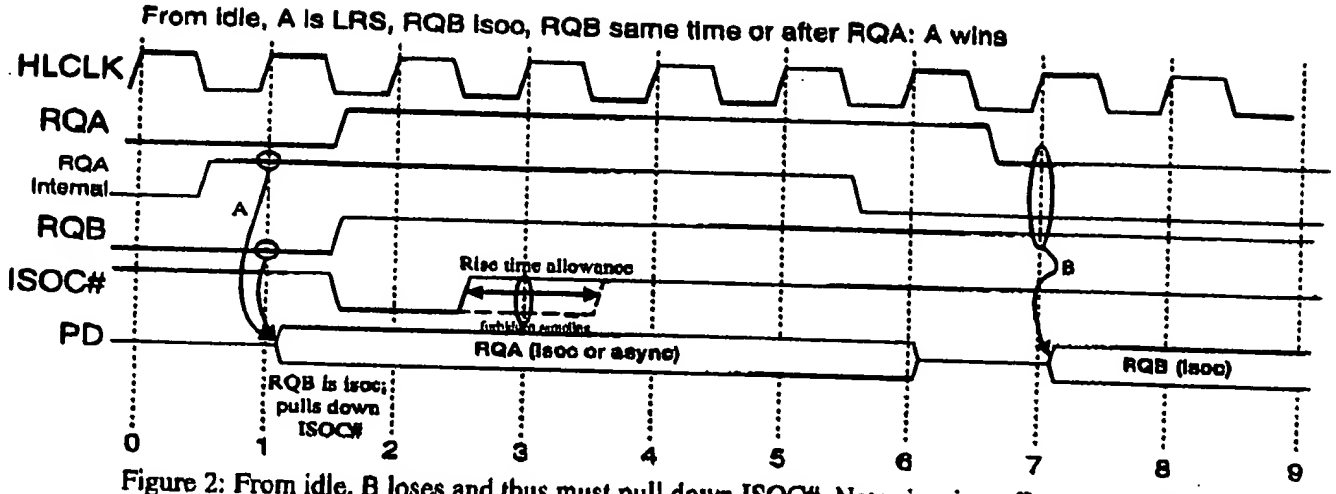

Figure 2: From idle. B loses and thus must pull down ISOC#. Note rise time allowance of wired-OR ISOC# signal.

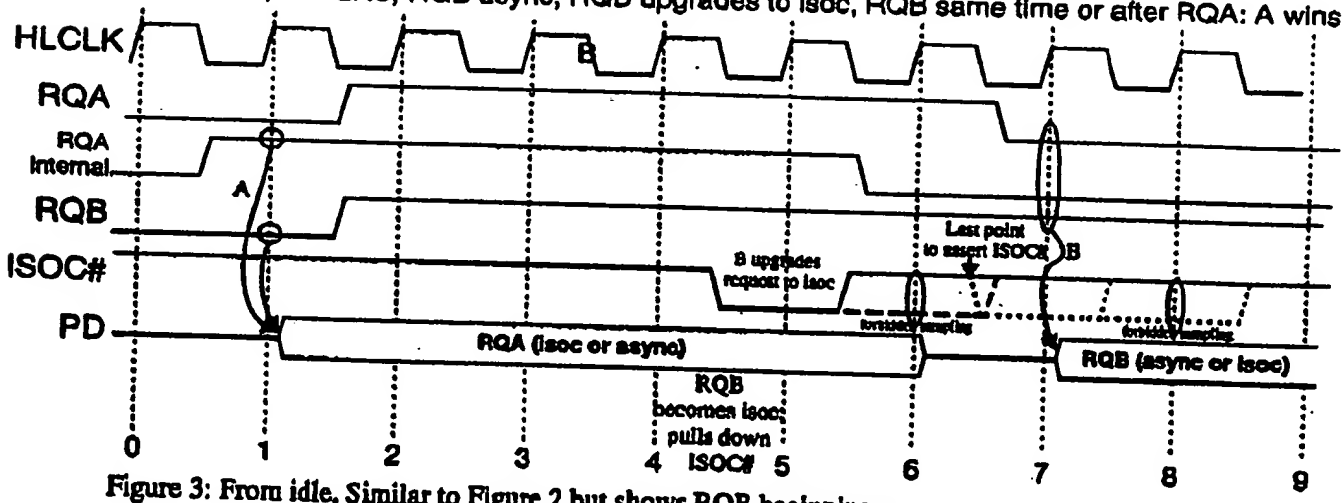

Figure 3: From idle. Similar to Figure 2 but shows RQB beginning as an async request and then upgrading to isoc.

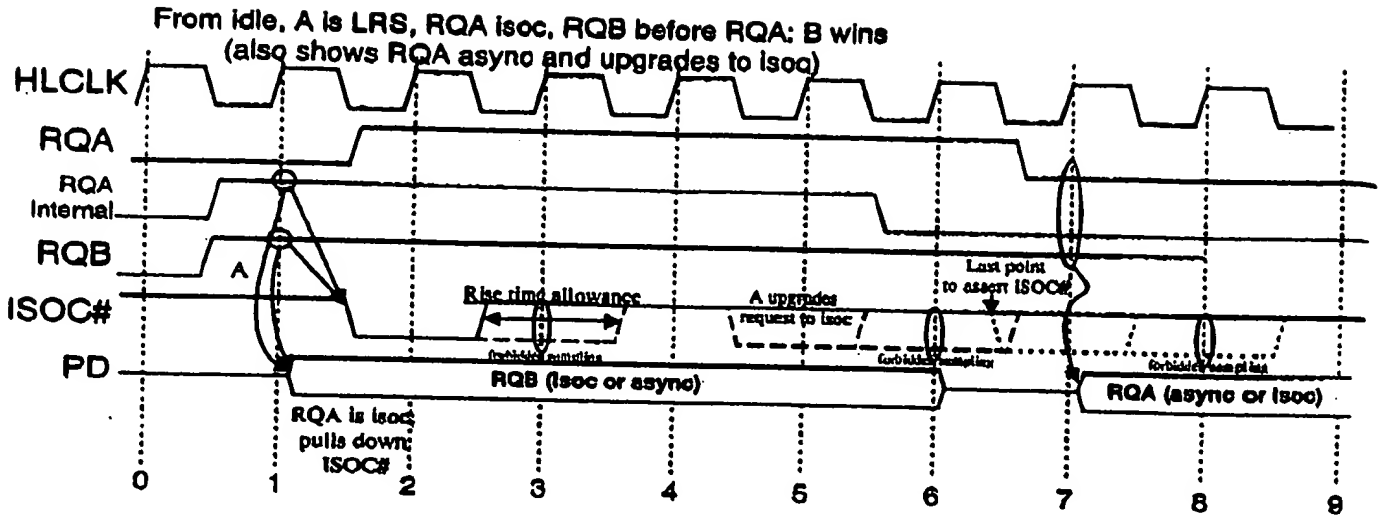

Figure 4: Similar scenario as in Figure 2 except this time B wins the grant.

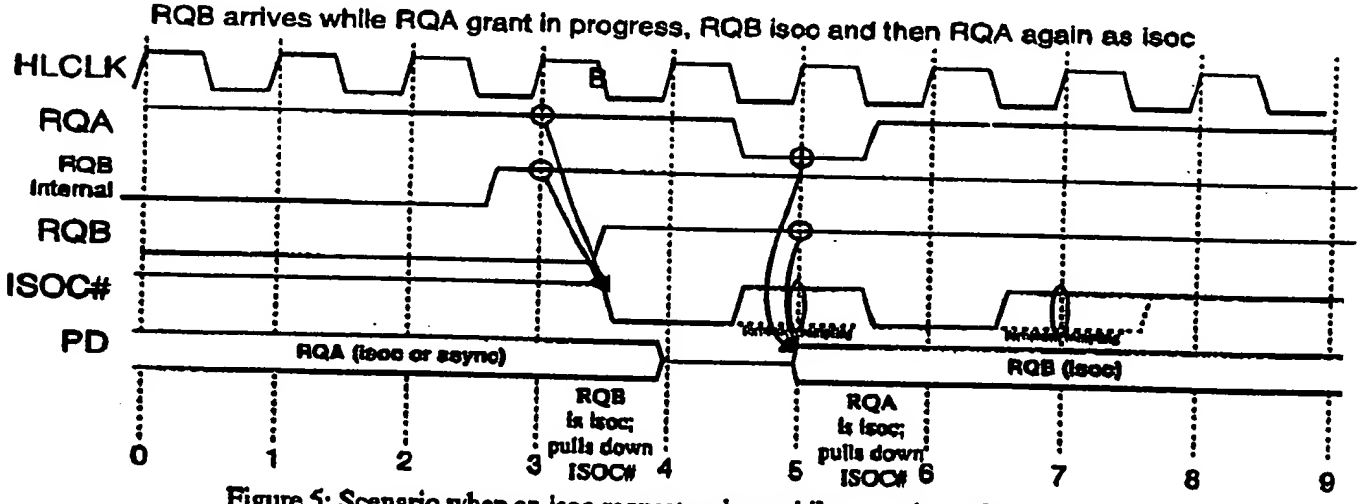

Figure 5: Scenario when an isoc request arrives while opposite end granted.

R Thorne
9/24/98

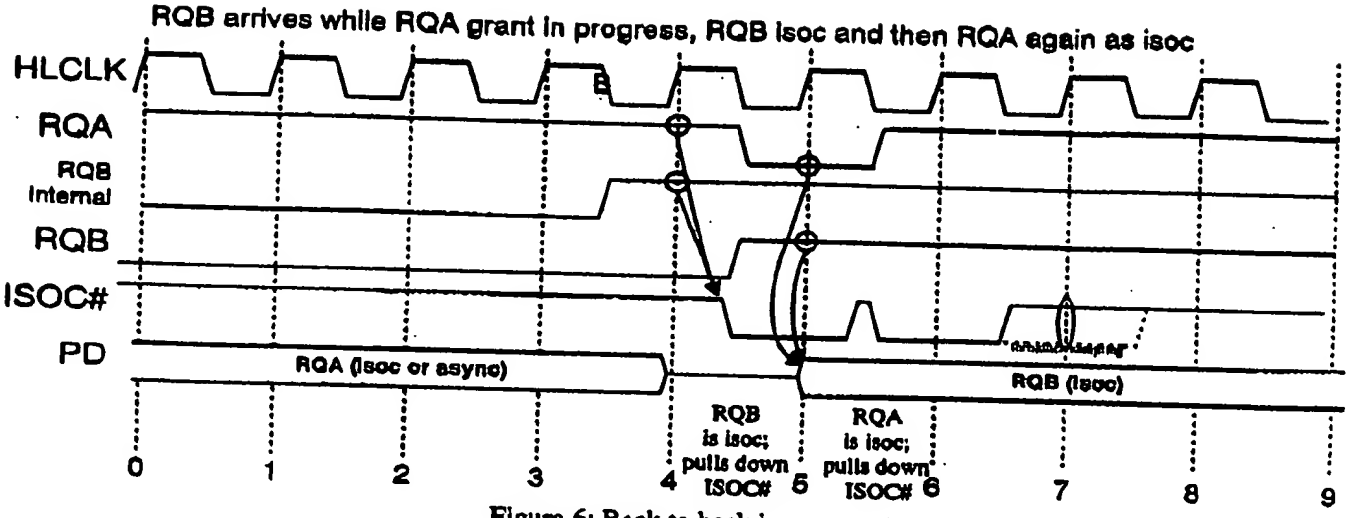

Figure 6: Back to back isoc requests

Figure 6 illustrates the main need for a wire-OR signal for ISOC#: back to back requests from one end concurrently with a request from the other end can lead to back to back assertions of the ISOC# signal by opposite ends. With wired-OR logic for the ISOC# signal there is no need to add complexity to manage the ownership of ISOC# and adding a clock for turnarounds. However, the rise time period can be violated – though with benign consequence. In Figure 6 if instead the RQA asserted in clock 5 was for async there would be back to back requests, first isoc and then async. Thus ISOC# would not be asserted at clock edge 6, yet end B would sample ISOC# at edge 6, in the middle of its rise time period following assertion by A at edge 5. The consequence of this rise time period violation is end B could incorrectly conclude that A had an isoc request. Since B is already isoc, the worst case is B's request downgrades to async and then withdraws its request to pre-empt in preference to the apparent isoc request from A. However, once A gets the grant it behaves as async. And if B then presents an isoc request, it will assert ISOC# and A will withdraw its async request.

Alternatively the rise time violation can be avoided by simply changing the protocol to defer assertion of ISOC# if ISOC# was asserted 1 clock before. In this case the RQA in clock 5 of Figure 6 would behave as an async request that is upgraded to ISOC# in 1 clock. This change would permit ISOC# to be an ordinary common clocked signal. In this case, an end must sample ISOC# and if already asserted, must wait one clock for turnaround before that end may assert its ISOC#. This is the idea behind the pulsed and duration versions displayed in the next subsection.

3.2.2 Pulsed and duration versions

The pulsed and duration versions are identical except for the signaling duration on the ISOC wire. Hence, the Figures below only illustrate the pulsed version.

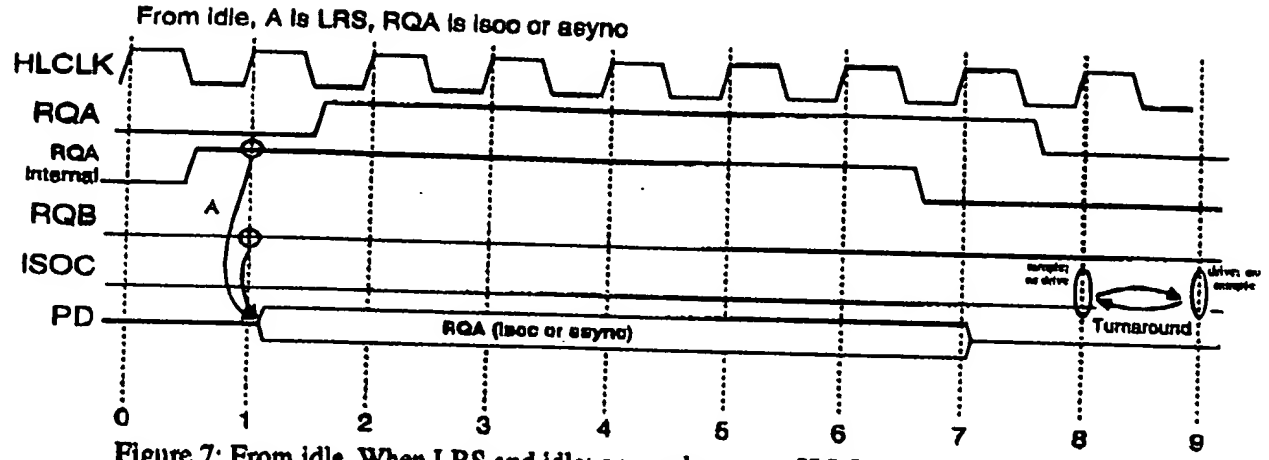

Figure 7: From idle. When LRS and idle; no need to assert ISOC. Note turnaround: ISOC is sampled by B, but not driven by A on clock edge 8 and then is driven but not sampled on clock edge 9.

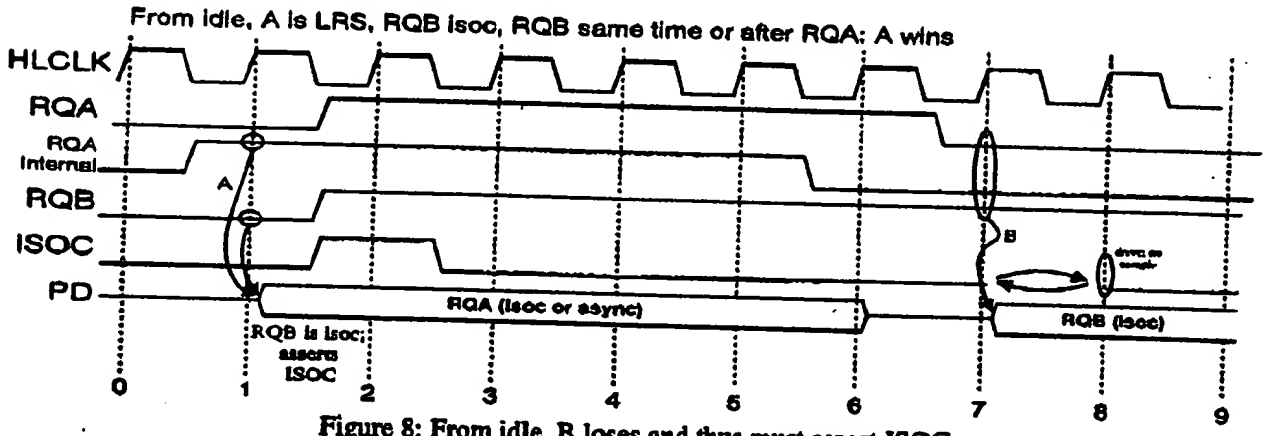

Figure 8: From idle. B loses and thus must assert ISOC.

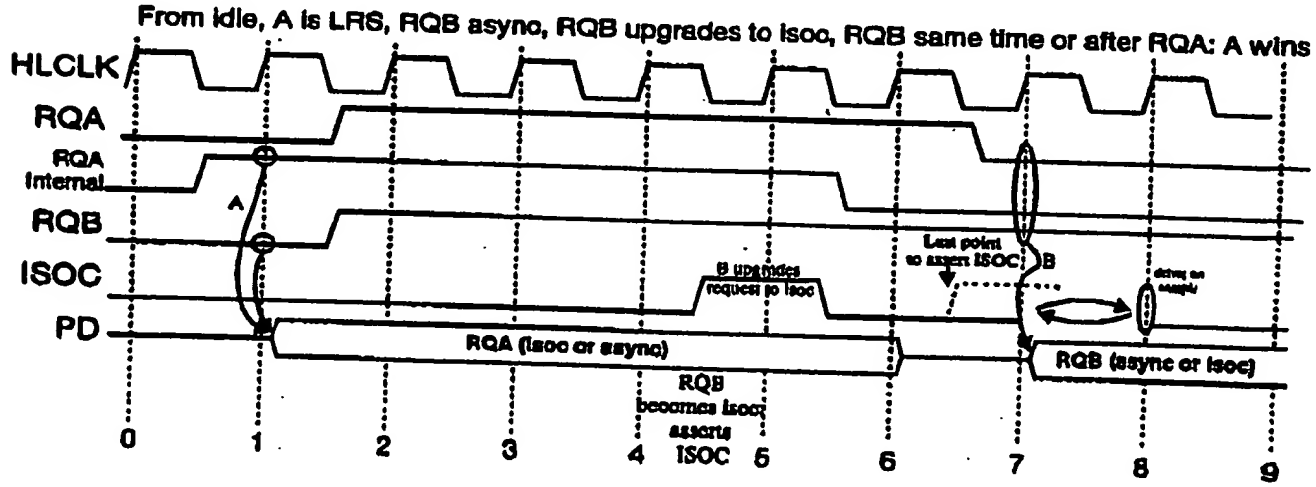

Figure 9: From idle. Similar to Figure 8 but shows RQB beginning as an async request and the upgrading to isoc

R. Ulbrne 9/24/98

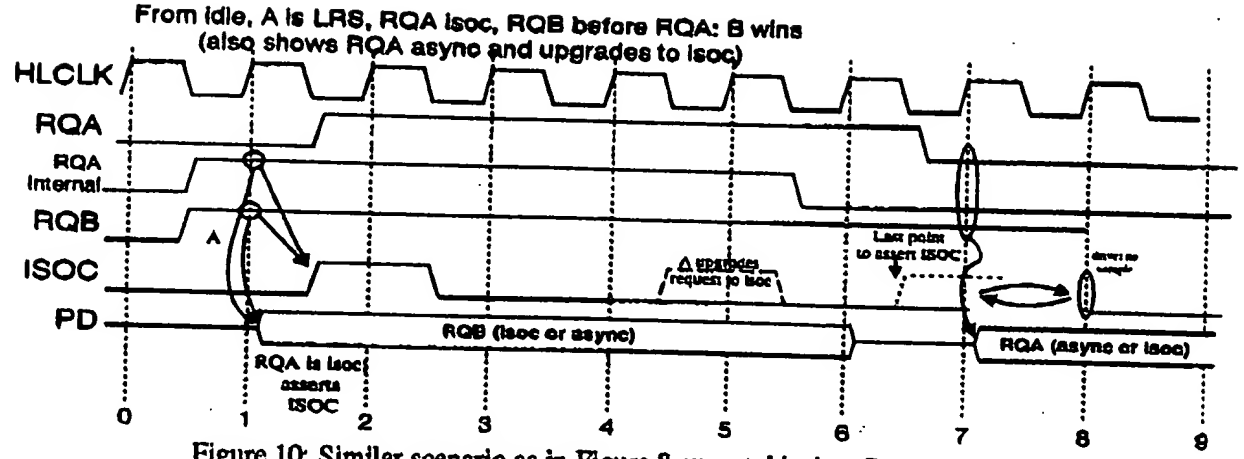

Figure 10: Similar scenario as in Figure 8 except this time B wins the grant.

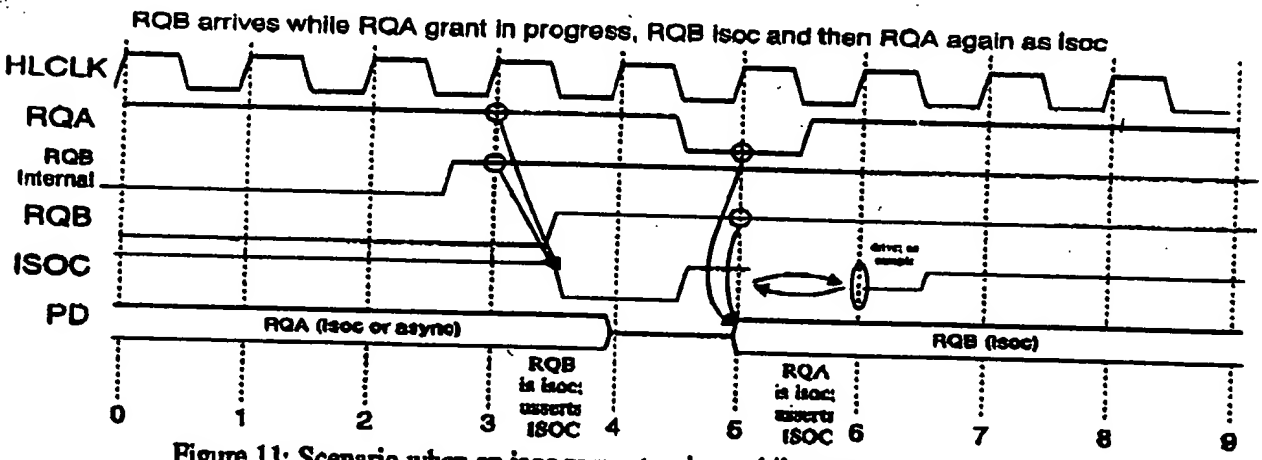

Figure 11: Scenario when an isoc request arrives while opposite end granted.

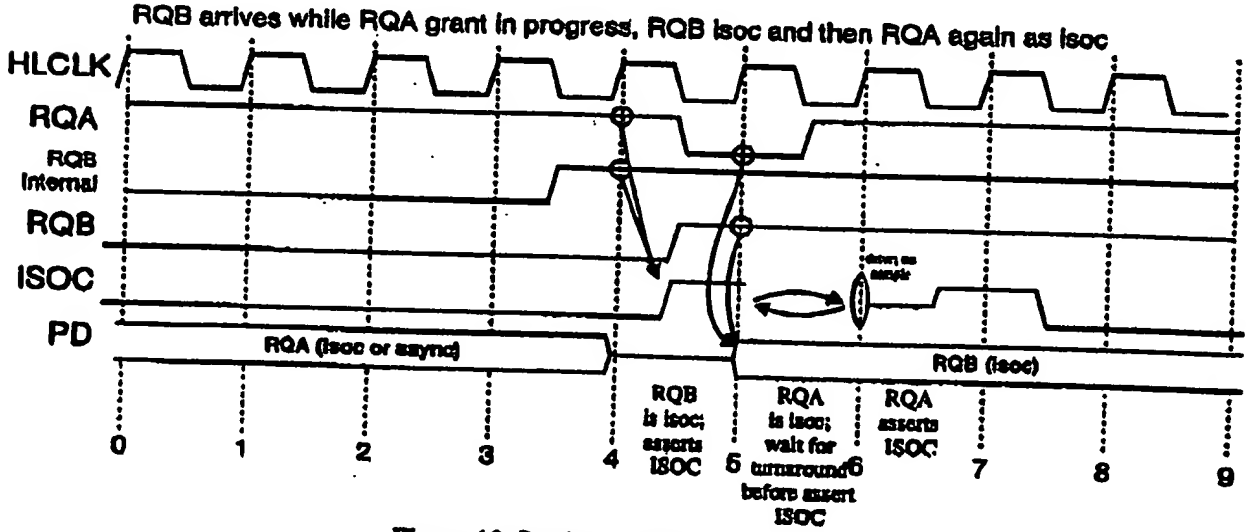

Figure 12: Back to back isoc requests.

9/24/98

In Figure 12, the turnaround of the ISOC wire forces end A to delay assertion of ISOC for one clock after RQA is asserted. This of benign consequence since it is equivalent to RQA starting as async and then upgrading to isoc. Note that end A must delay assertion of ISOC regardless of the type (isoc or async) of the ISOC wire sampled at clock edge 5. In the case of pulsed signaling and parallel termination, a slight optimization is possible. With parallel termination, the turnaround cycle must only be observed between consecutive assertions of ISOC by opposite ends. Thus delaying the assertion of ISOC as shown in Figure 12 only needs to occur if ISOC was asserted at edge 5 and RQA presented at edge 6 is isoc. Otherwise, end A can assert ISOC concurrent with RQA for edge 6.

3.3 Benefits and Generalizations

There is no degradation of async performance in the absence of isoc traffic. In an idle system, the latency of either isoc or asnyc requests is the same as in the present Hub Link.

This invention can be applied to the arbitration of any two classes of requests with different service requirements and can also be used to signal pre-emption from one end to the other.

This invention can be extended for half duplex buses other than Hub Link in which there may be no concept of LRS and MRS. The main idea in such a general environment is that each end pulls down a wired OR signal ISOC# whenever that end's request is isoc and any ambiguity about which end that pulled down ISOC# is resolved in a two phase arbitration. If the REQ lines indicate that only one request is present, then arbitration proceeds as in the present Hub Link and ISOC# is effectively ignored. However, if two requests are present and ISOC# is asserted, then each end executes a two phase protocol. In the next clock each end removes its request if it is async and then re-samples the REQ lines the following clock. Either one or two requests may remain at this following clock; but all such requests must be isoc. If one isoc request remains, it is granted to give isoc traffic priority in use of the Hub Link over async. If two isoc requests remains, the arbiter makes a choice, such as granting the end opposite from the last isoc request serviced (least recently used).

Figure 13 shows an example of the two phase protocol

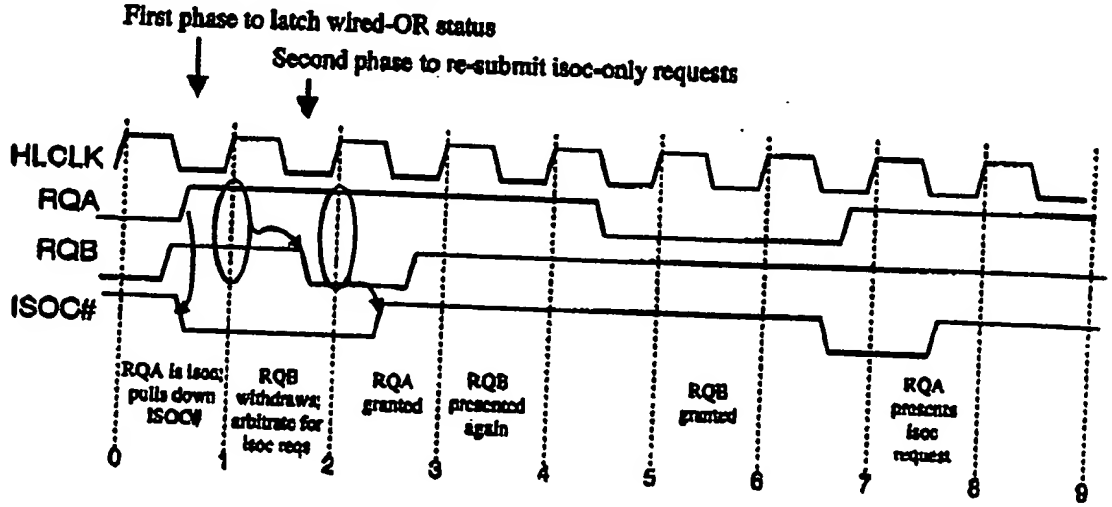

Figure 13: Two phase wired-OR protocol

9/24/98

4. Value of Invention to Intel

This invention is proposed for addition to the Hublink specification. Hublink is a key technology for interconnecting chips in Intel chipsets and later, Intel processors to I/O chips. If accepted, this invention may be used in Intel chipsets such as the McKinley/Foster chipset to improve the isoc performance of the IOH to ICH Hublink and later in Timna2 to connect the processor and I/O devices.

In addition, should Intel decide to open the Hublink interface to other chip vendors, intellectual property covering Hublink and various optimizations to it will be important. Any implementation details yielding superior performance is a competitive advantage.

5. Infringement

If Intel decides to open the Hublink interface, other vendors building chipsets may improve or optimize upon Intel's Hublink specification. In so doing they might use the subject invention. Infringement could be detected by the addition of an additional pin used to signal additional request information to the opposite end.